MODELING PILOT PERFORMANCE USING AUTOMATED AIRCRAFT SYSTEMS

George Mason Research Team

January 2002

GMU

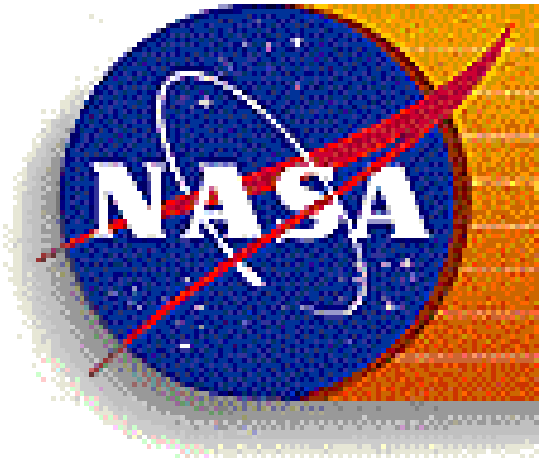
Project Team

- George Mason University
  - Deborah A. Boehm-Davis
  - Ronald Chong
  - Melanie Diez
  - Jeffrey Hansberger
  - Robert W. Holt
  - Mary Pinney
  - Wolfgang Schoppek
- Consultants:
  - Captain Bill Hamman
  - Lance Sherry
  - Captain Frank Tetreault

GMU

Project Support

Office of the Chief Scientific and Technical Advisor for Human Factors: AAR-100



NASA Ames Research Center
Aviation Systems Research Technology & Simulation Division

GMU

Overview

- Model pilot performance
  - Understand cognitive processes underlying pilot performance in automated cockpit
- Single pilot Model
  - Single pilot model interacting with dynamic automation system
- Crew Model
  - Captain and First Officer pilot models “communicating” to each other

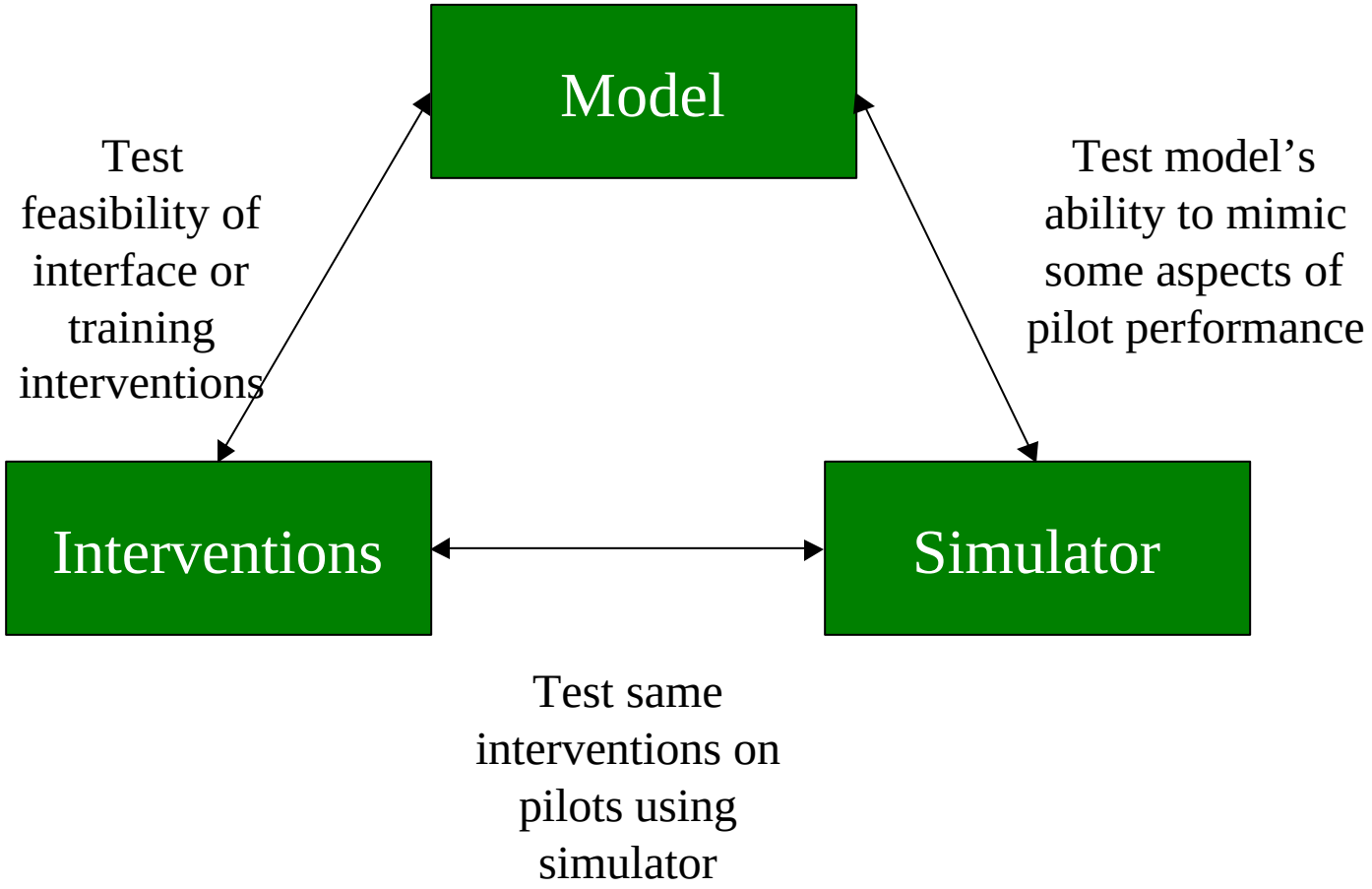
Modeling a single pilot system

- Context: Single pilot operating an automated commercial aircraft during descent phase of flight
  - Dynamic environment
    - Air Traffic Control commands
    - Required changes in path, altitude, airspeed, and aircraft configuration during descent
  - Complex device
    - Advanced automation system with rich information displays
    - Required programming tasks, operating mode selection, etc.
- Modeling Focus: Pilot use of automation modes during descent
  - Vertical Navigation, Vertical Speed, Flight Level Change

Modeling Goals

- Understand cognitive processes underlying pilot performance in automated cockpit
- Use that information to develop interventions to improve performance
- Evaluate effect of interventions using the model

Cognitive Modeling Approach



GMU

Task characteristics of "Flying a highly automated aircraft"

	"Flying"	*"Typical" ACT-R Task*
Example Task	Flying down from cruise altitude to final approach fix with several altitude and speed restrictions	Memorizing lists, Judging statements, Tower of Hanoi
Time scale	Minutes to hours	Seconds to minutes
Dynamic	Environment changes rapidly and autonomously	Environment is relatively static
Goal structure	Heterogeneous goals	Single main goal & sub goals

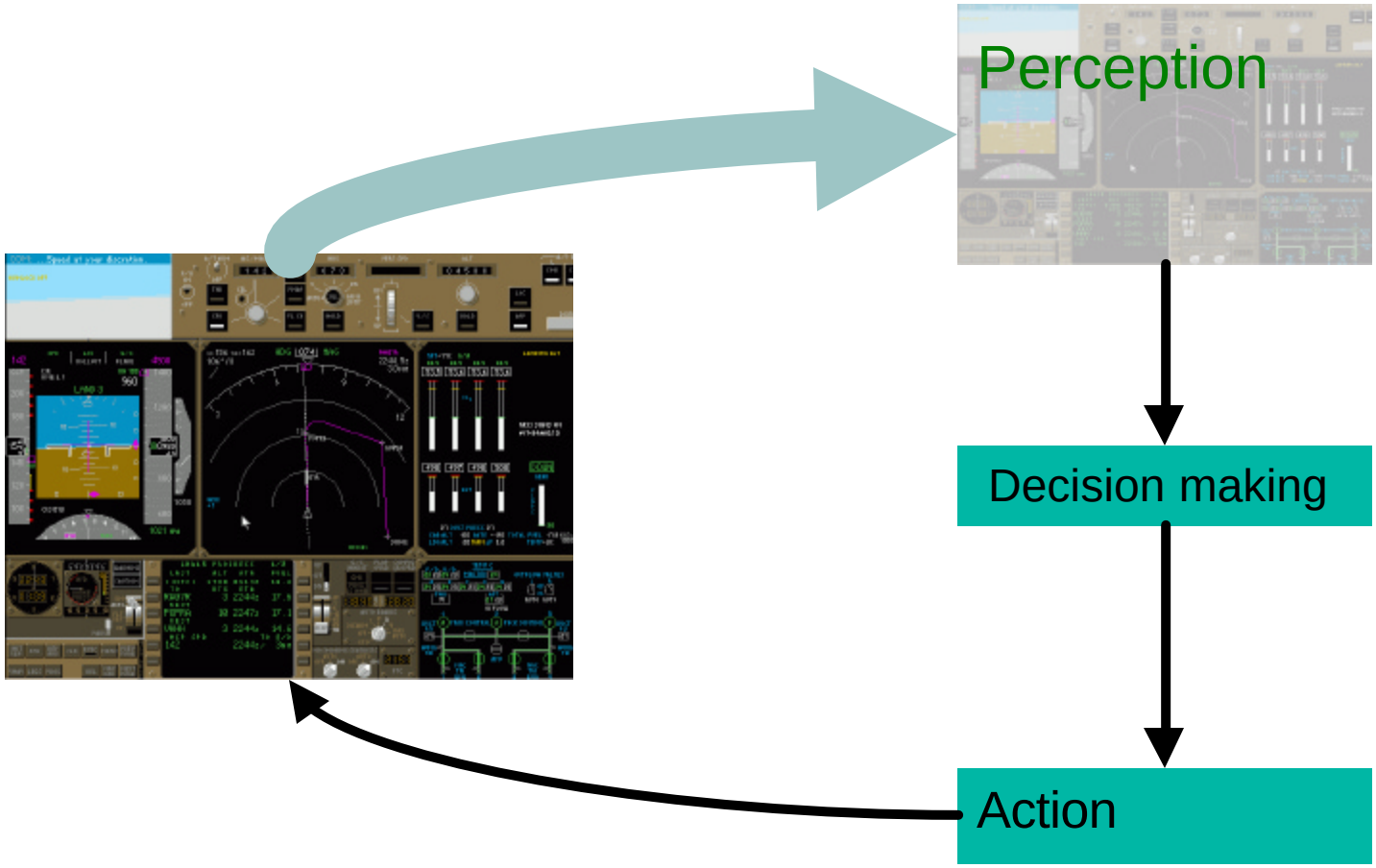
⇒ New solutions have to be found to cope with these task characteristics

GMU

Aerowinx B747 Desk-top Simulator Interface

GMU

Simulator and Single Pilot Model



Representation of Procedural behavior and cognition: ACT-GOMS

- "Translation" of NGOMSL to ACT-R 4.0
  - Memory representation for GOMS-level elements
  - New elements (methods, steps, operators, desired states)
- Added features:
  - Control structure (handling operational goals, intentions, and interruptions)
  - Goal stack limited to 3 levels (shallow)
  - Activation-based retrieval of goals and steps

Example for NGOMSL Level

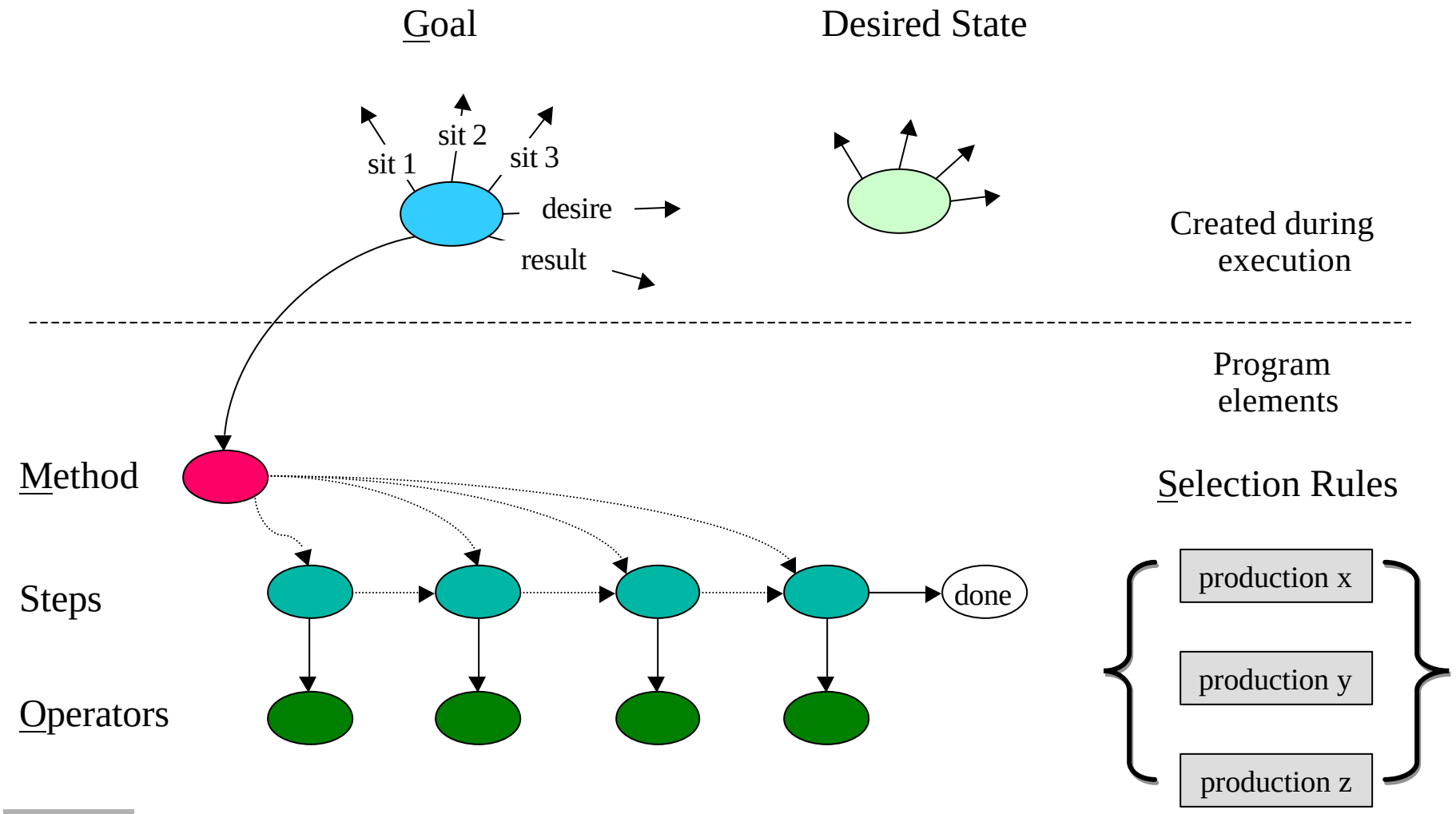
Encode Clearance ...
Decide Descent Method
retrieve target altitude
enter MCP altitude
SR: Waypoint in clearance? (yes)

Calculate S
get altitude
encode altitude
retrieve target altitude
encode target altitude
subtract the two values
encode result

FLCH Descent
press FLCH
wait (5 seconds)
retrieve target waypoint
check-green-arc
SR: green arc aligned with waypoint? (no)
SR: too steep? (yes)
decrease MCP speed
wait (5 seconds)
go to step 4

GMU

ACT-GOMS details



Goals

Desired State

(represents the target state)

```
(chunk-type d-state
        type
        p1
        p2
        p3
        p4
        parent)
```

Step-specific goals

Main Goal Type

(created to carry out a method)

```
(chunk-type main-goal
        mode
        s1
        s2
        s3
        desire
        result
        method
        step-type
        step
        operator)
```

Basic Goal

(always on bottom of the goal stack)

```
(chunk-type basic-goal
        rehearsal-chunk
        focus-chunk
        n
        result)
```

Operators

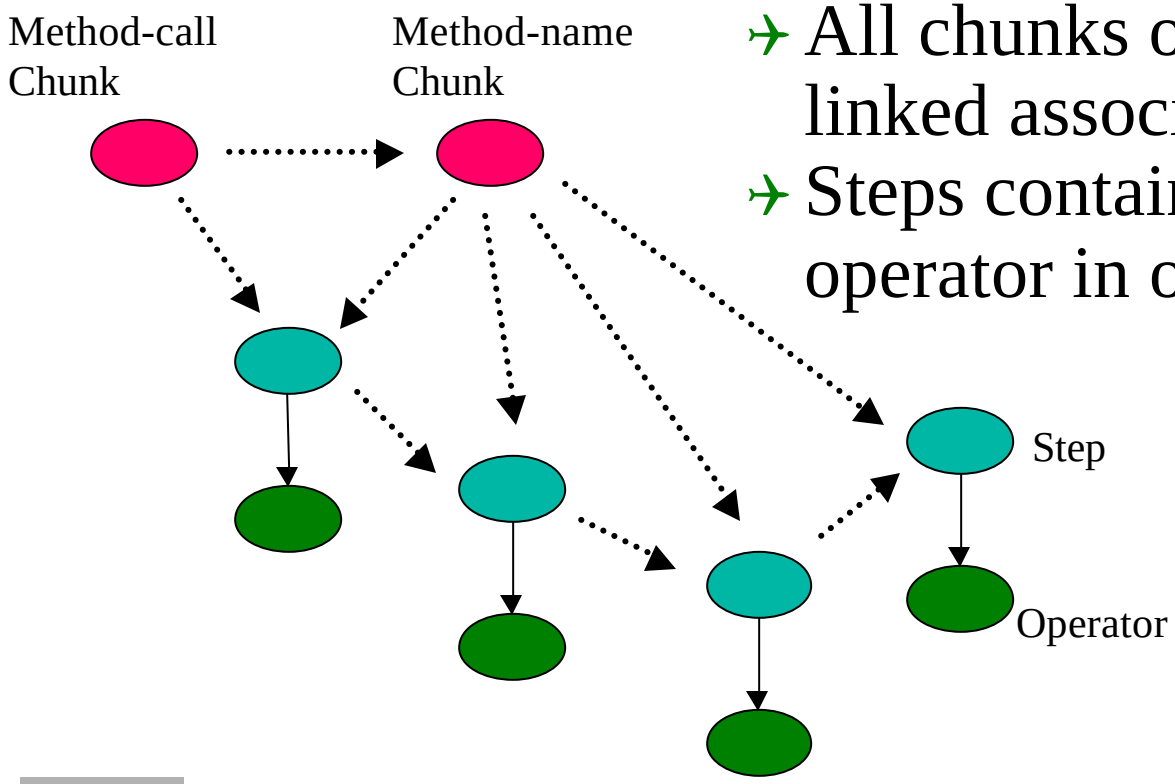
Each step in a procedure uses 1 operator (unless it is a step calling a new method)

3 basic types of operators are used

- Internal operators
  - mental calculations, comparisons, memory operations, etc.
  - one production per internal operator
- External operators
  - pushing buttons, entering values, etc.
  - fixed sequence of generic productions to execute the operation
- Perceptual operators
  - reading displays, verifying displayed values, etc.
  - fixed sequence of generic productions to obtain the perception
  - each perceptual step results in the creation of an episodic representation of the perceived value

Methods

- Methods are represented as a method-call chunk, a method name chunk, and a number of steps
- All chunks of the group are linked associatively (S_{ji} values)
- Steps contain the name of an operator in one slot



GMU

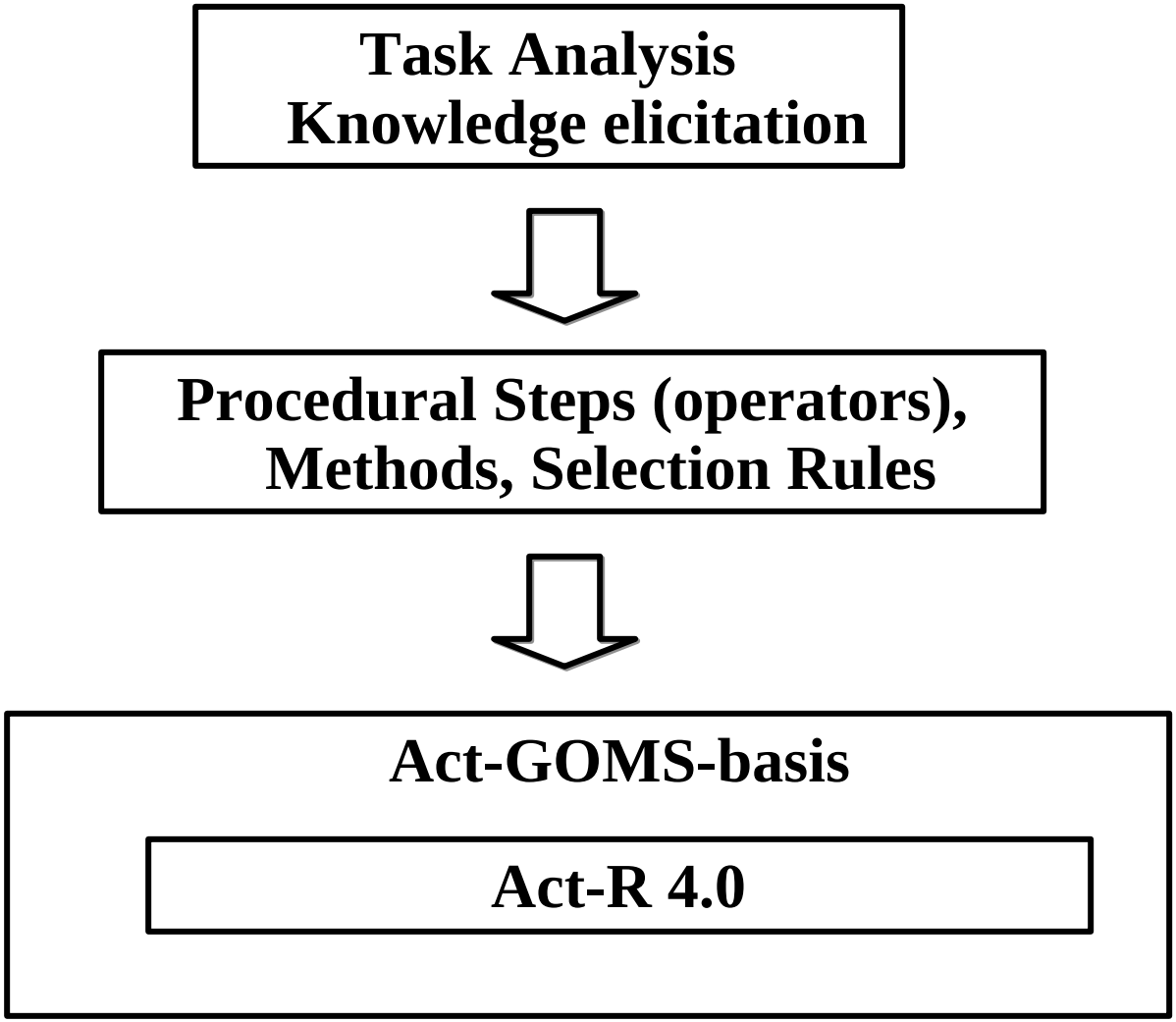
Selection Rules

- Selection rules are represented as steps in a method; they trigger certain productions[1]
- Selection rules determine:
  - branching: which methods to try to activate next
  - termination of methods that have a termination condition

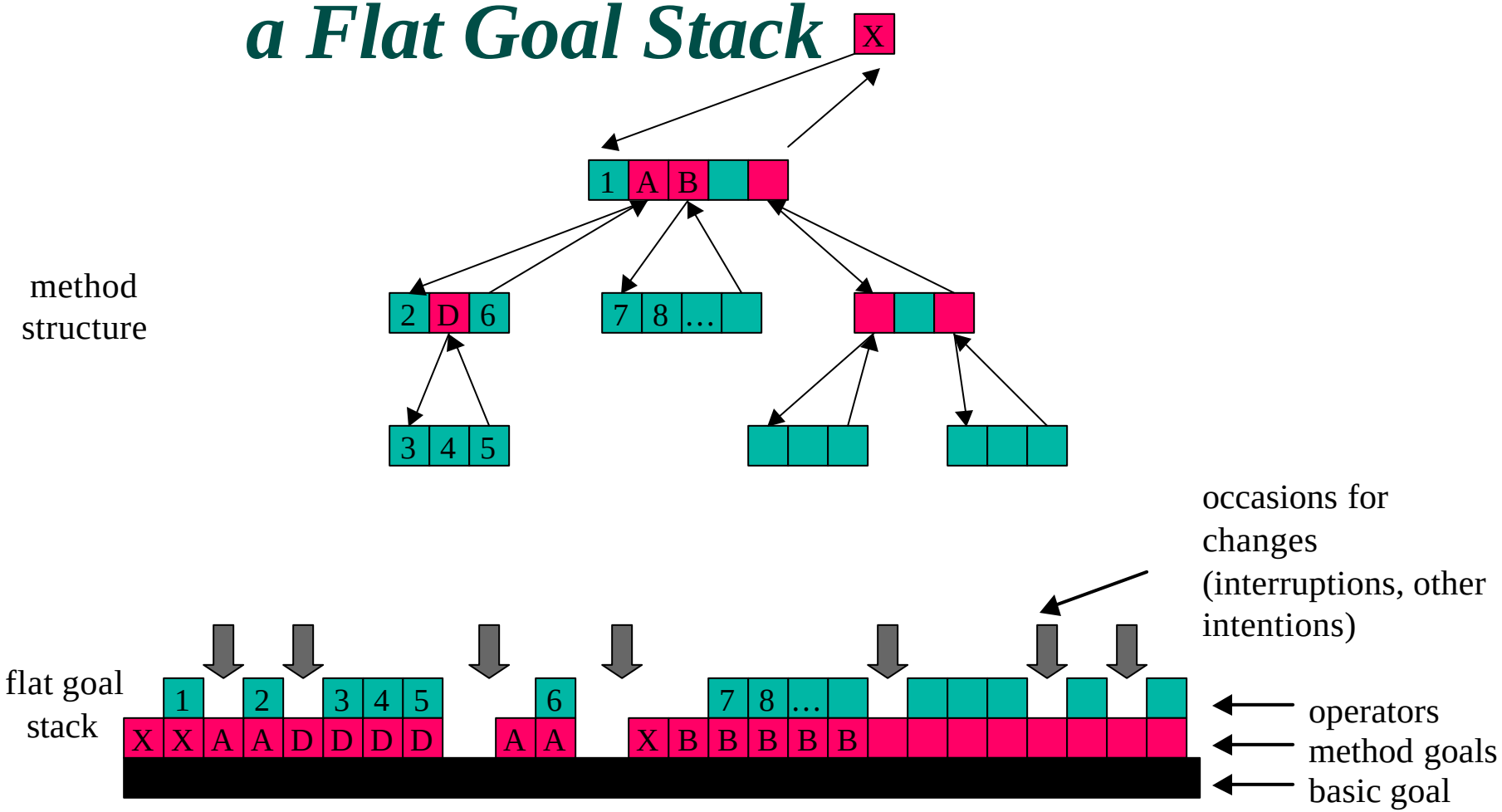
```
(p decide-to-use-vnav
  =goal>
     isa main-goal
     step-type 'sr
     method n-calcs1
     result =s
  !eval! (<= =s 250)
  =method>
     isa method
     name n-vnav
 ==>
  =goal>
     isa main-goal
     op =method
     step-type 'meth
     result nil)
```

[1] very much like an internal operator, ☞ Operators

Current Approach to model building



Hierarchical Methods executed with a Flat Goal Stack



(cf. Altmann & Trafton, 1999)

Goal Operations with Deep vs. Flat Goal Stack

	Deep goal stack	*Flat goal stack*
On begin of new method	push new goal	rehearse current goal, replace it with new goal
On end of method	pop old goal	pop old goal, retrieve next goal from memory[1]

[1] Next goal can be part of "the plan" but also an intention or an interruption.
⇒ This makes the model flexible, but also vulnerable to procedural errors.

Get Goal

```
(p get-goal
  =goal>
     isa basic-goal
     result =result
     reh-chu nil
  =other-goal>
     isa main-goal
     mode =mode
 ==>
  =goal>
     foc-chu nil
  =other-goal>
     mode nil
     result =result
  !push! =other-goal
  !output! ("Found ~A, Mode ~A" =other-goal
   =mode)
)
```

Triggers when goal stack has only the basic goal on it

Selects most active available main-goal.

Retrieval competition: No symbolic linking

Pushes main-goal as level 2 on stack

Get Step

```
(p get-step
  =goal>
     isa main-goal
     step-type 'next
     op =op
     et =et
  =step>
     isa step
     type =type
     orga =orga
   - operator =op
     operator =on
 ==>
  =goal>
     step =step
     op =on
     step-type =type
     next =orga
  !output! ("retrieved ~S" =on)
)
```

Triggers when goal stack has a main goal on top which requires a next step

Selects most active available step of any type.

Retrieval competition: No symbolic linking

The step slot in the main-goal is set to the name of the step

GMU

Advantages of Associative Linking of Steps

- Occasional step skipping during performance, particularly under high working memory load
- Deviations from a strictly linear sequence of steps are possible (e.g. shortcuts)
- No special learning mechanism needed (ACT-R associative learning does the job)

$\Rightarrow$ A more realistic representation than symbolic linking?

Lessons Learned from the Single Pilot Model

- Unique features of single pilot model
  - GOMS-level approach implemented
  - Flat goal stack
- Issues
  - Model performs perfectly OR gets lost
  - Learning of S_{ji}s caused problems

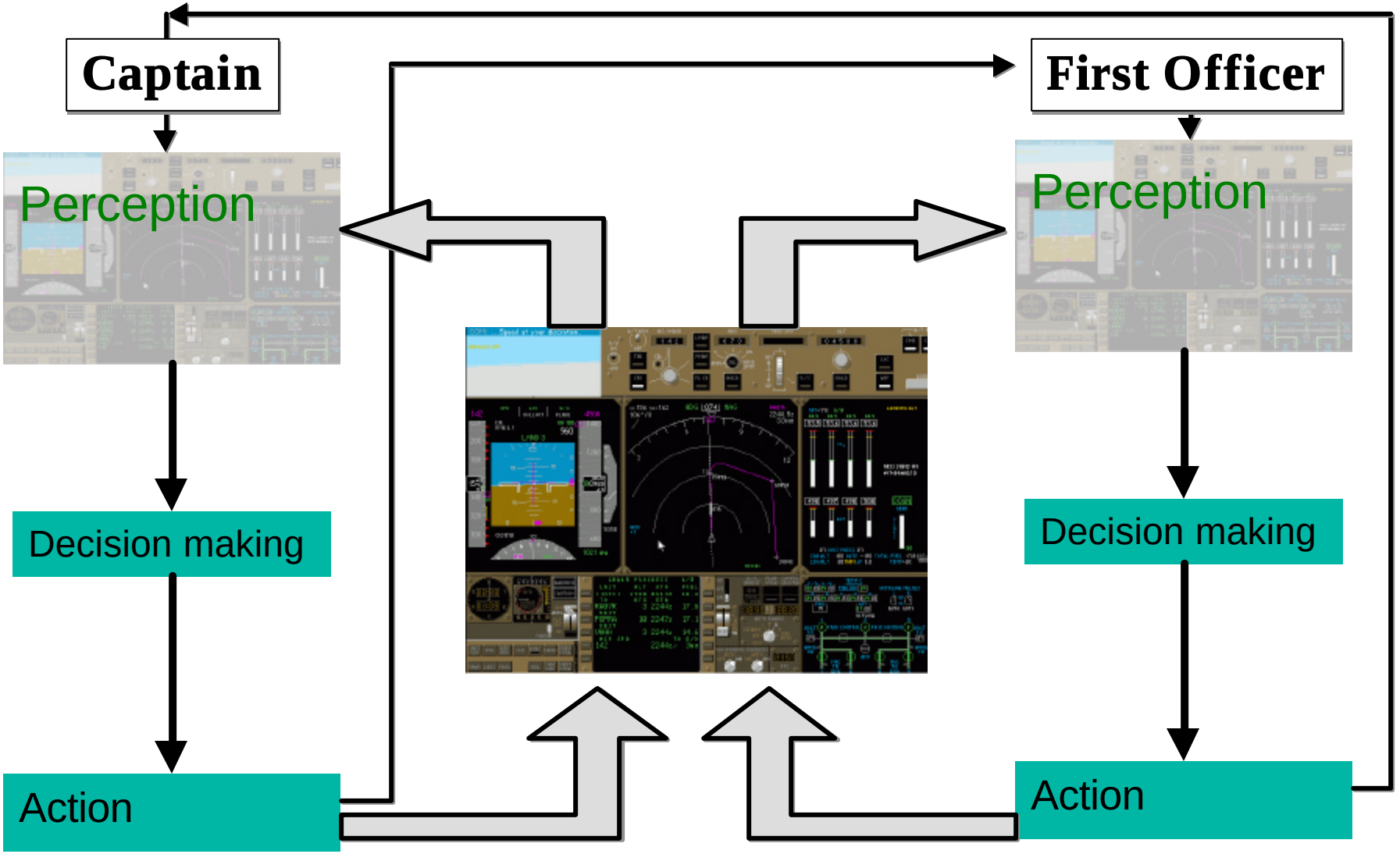
Modeling the crew system

- Context: Two-pilot crew operating a commercial aircraft during descent phase of flight
  - More complete task analysis
    - Large hierarchy of linked goals with lower-level steps
    - Scripted by checklists and Flight Operations Manual
  - Two-pilot crew
    - Pilot Flying and Pilot Not Flying task division
    - Communication between pilots (and ATC)
- Modeling Focus: The communication and actions during descent
  - Two individual pilot models talking with each other
  - Leveraged from single-pilot automation model

Modeling Goals

- Model crew automation interaction
  - Explore effects of specific aspects of crew experience and workload on simulated task performance
- Improve assessment of real crew performance based on model results
- Applications:
  - Training
  - Proficiency evaluation

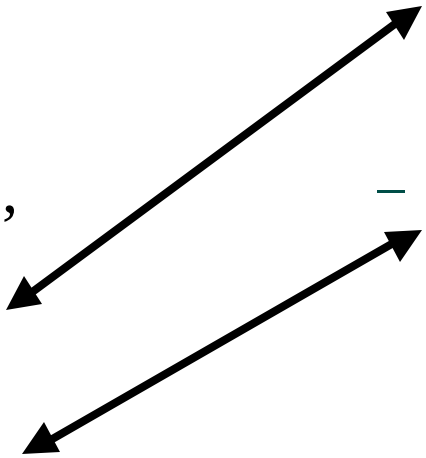
Simulator and Crew Model



GMU

Constructing a crew model

- PF model
  - Receive ATC clearances
  - Decide on descent mode of FLCH, V/S, or VNAV
    - » change mode of descent when necessary
  - Monitor A/C status, weather, traffic...
  - Divide other flight tasks with PNF
  - Communicate with PNF



- PNF model
  - Do appropriate checklists
    - » Program FMS for descent
    - » Other checklist tasks
  - Do other flight tasks
    - » get / set radio freq.
    - » approach plates, etc.
  - Communicate with PF
    - » Required communication such as required briefings
    - » Optional communication
  - Communicate with FA, PAX, etc.

GMU

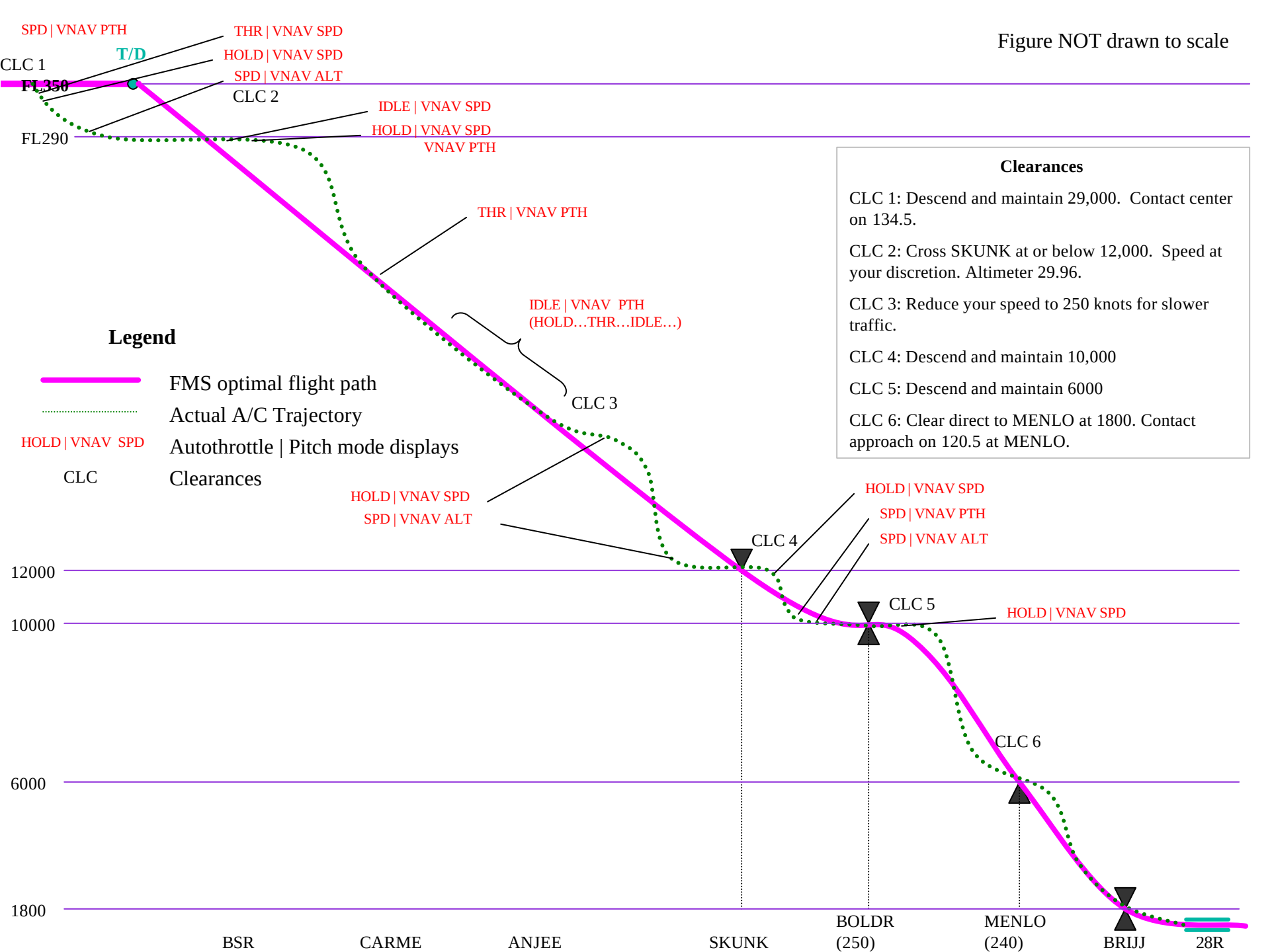


Figure NOT drawn to scale
SPD | VNAV PTH
THR | VNAV SPD
T/D
HOLD | VNAV SPD
SPD | VNAV ALT
CLC 1
FL350
CLC 2
IDLE | VNAV SPD
HOLD | VNAV SPD
FL290
VNAV PTH
THR | VNAV PTH
IDLE | VNAV PTH
(HOLD…THR…IDLE…)
CLC 3
Clearances
CLC 1: Descend and maintain 29,000. Contact center on 134.5.
CLC 2: Cross SKUNK at or below 12,000. Speed at your discretion. Altimeter 29.96.
CLC 3: Reduce your speed to 250 knots for slower traffic.
CLC 4: Descend and maintain 10,000
CLC 5: Descend and maintain 6000
CLC 6: Clear direct to MENLO at 1800. Contact approach on 120.5 at MENLO.
Legend
FMS optimal flight path
Actual A/C Trajectory
HOLD | VNAV SPD
Autothrottle | Pitch mode displays
CLC
Clearances
HOLD | VNAV SPD
SPD | VNAV ALT
CLC 4
HOLD | VNAV SPD
SPD | VNAV PTH
SPD | VNAV ALT
12000
10000
CLC 5
HOLD | VNAV SPD
CLC 6
6000
1800
BSR
CARME
ANJEE
SKUNK
BOLDR
(250)
MENLO
(240)
BRIJJ
28R

Overview of Model Structure

- Uses the ACT-GOMS representation of procedural knowledge
- Procedural behavior based on Single Pilot Model (ACT-R 4.0)
- Does not currently interact with a real simulator
- Task analysis based on checklists and flight operations manual of a major airline; cognitive analysis carried out with SMEs
- S_{ji}s specified a priori rather than learned

Pros and cons of ACT-GOMS

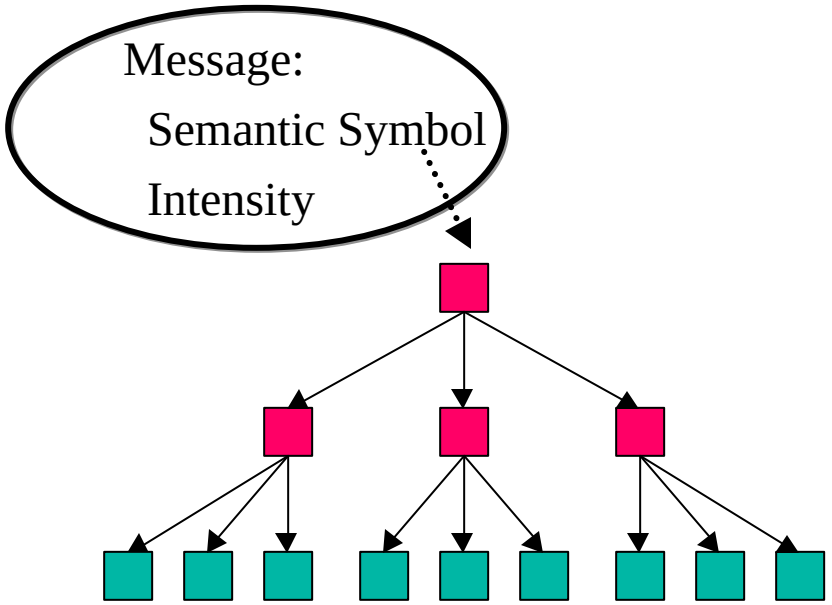
- Pros
  - Direct translation of task analysis to methods and steps in the model
  - Natural production of certain qualitative results:
    - Omission errors: e.g. Step-skipping
    - Commission errors: e.g. intrusions from other sub-tasks
    - Factors influencing procedural performance:
      - "Expertise effects" (higher Sji s)
      - Higher goal activation
      - activation noise
    - Interruptions by other tasks
    - Forgetting of goals over long time intervals
- Cons
  - Not parsimonious
    - Debugging difficult: ACT-GOMS <-> ACT-R <->Lisp
  - More parameters to set and adjust to fit human data

GMU

Issue 1: Communication between models

- ACT-R *MultiModel* extension provides the “pipe” between the models
  - Still under development
  - “Speaker” model creates a 2-tuple message:
    - a semantic symbol; e.g. Put-down-flaps
    - an intensity; an integer value used to influence the base-level activation of the message chunk in the “listener” model.
  - “Listener” model polls its MultiModel input buffer (“Did I hear something?”) on every cycle.
  - On receiving a message, a goal chunk is created and is rehearsed the number of times indicated by the intensity.
  - Recognition of the goal/message chunk is subject to activation-based retrieval through Get-Goal

Issue 1: Communication (continued)



Communication message creates a goal to execute a pre-existing hierarchy of methods and steps.

But message could also specify a change-of-order from the normal procedure.(e.g. gear before flaps)

Implementing different types of messages may be necessary.

A complete version of message transmission may require some version of natural language processing.

Issue 2: Goal decay and "death"

- Our solution is periodic monitoring of external environment for goal cues followed by goal rehearsal
- Example Code

```
(p Refresh-Goal
    =goal>
       isa BASIC-GOAL
       reh-chu nil
       misc =rtime
     =most-active-undone-main-goal>
        isa main-goal
        mode =mode
      - step-type 'done
   !bind! =current-time (actr-time)
   !eval! (< =rtime (- =current-time *monitor-cycle-time*))
==>
   =goal>
      misc =current-time ;resetting the time
  !eval! (rehearse-chunk-fct (list =most-active-undone-main-goal)
  :repeat *checkpoint-rehearsals*)
  !eval! (mod-chunk monitor-environment method n-monitor step
  nil-c op m-monitor step-type 'next)
  !eval! (push-goal monitor-environment)
 )
```

Triggers when goal stack has only a basic goal and sufficient time has elapsed from the last refresh cycle.

Rehearses most recent main goal and pushes goal to check environment for goal cues.

Issue 3: Transition to ACT-R 5.0??

- Want the group's feedback and input here!
  - Embodiment of cognition and buffers is a big plus
  - What are the implications of the other changes in the architecture?
  - How will we treat the S_{ji}s? S_{ji} learning?

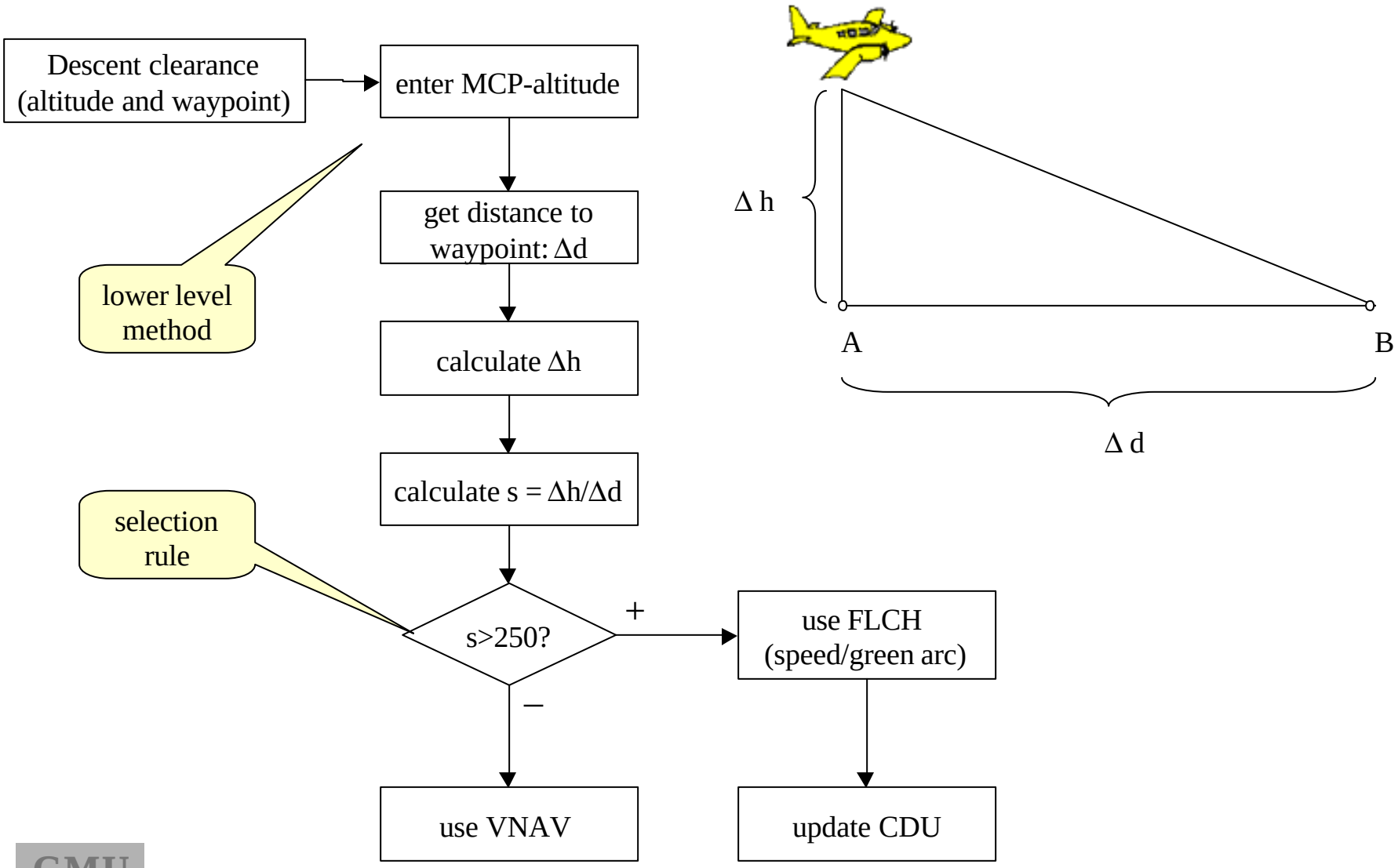
Summary

- We developed a layer for ACT-R 4.0 for handling procedural behavior and cognition
  - Associatively linked Methods and Steps
  - Shallow goal stack with competitive retrieval
- Approach applied to single-pilot use of automation during descent
  - Generalized to development of a crew model

Discussion?

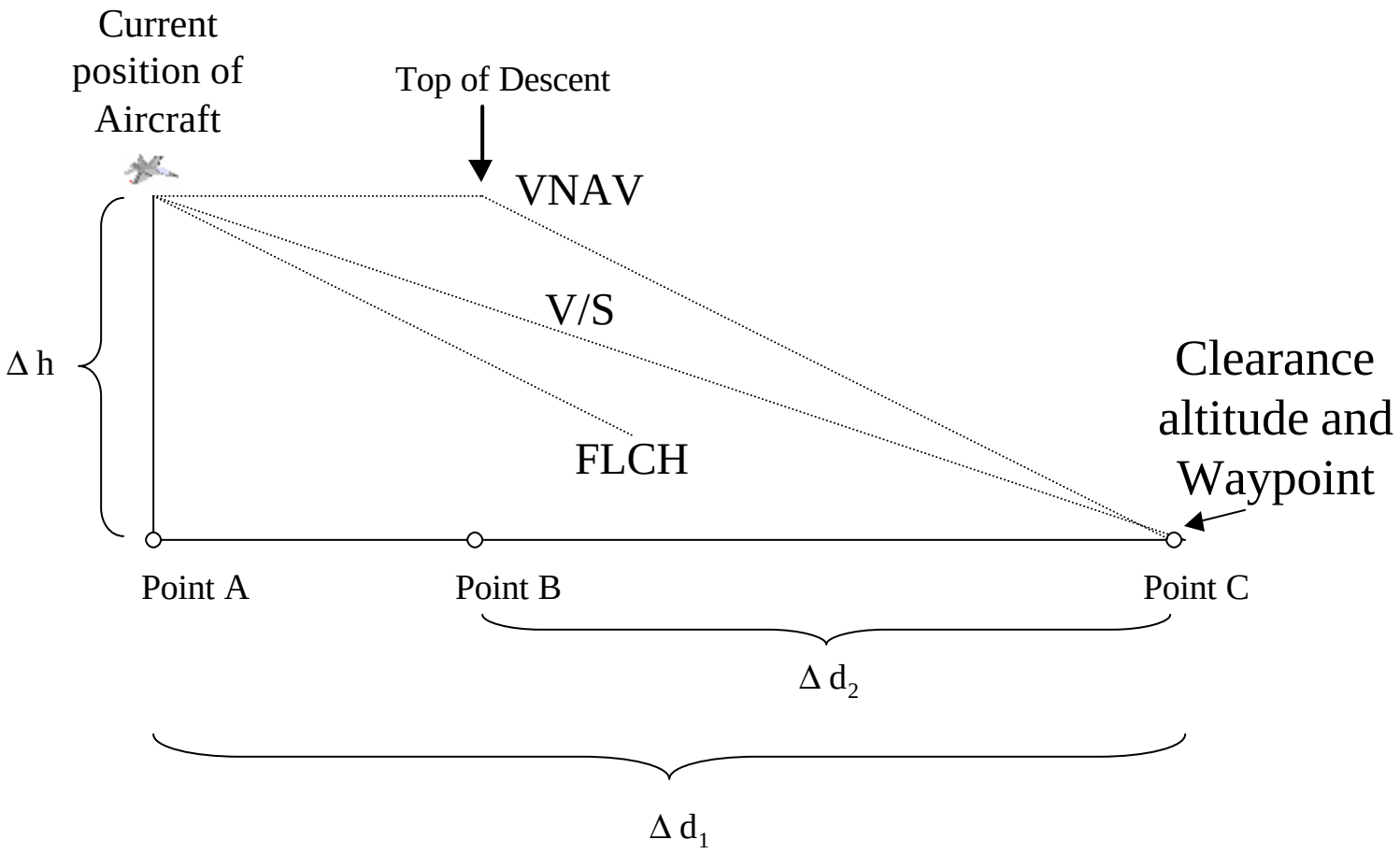
GMU

Example High Level Method

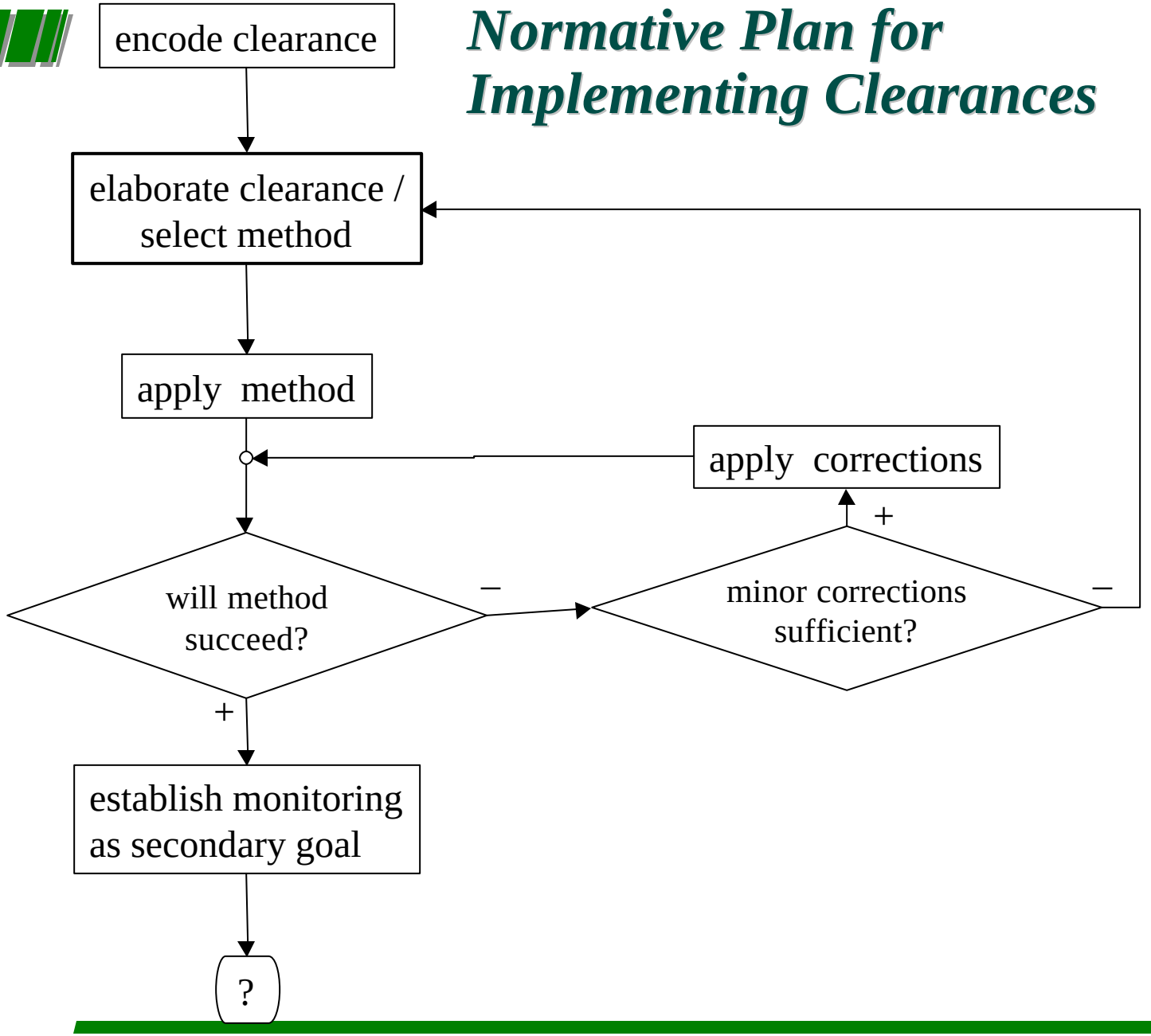


GMU

Pilot's Descent Options

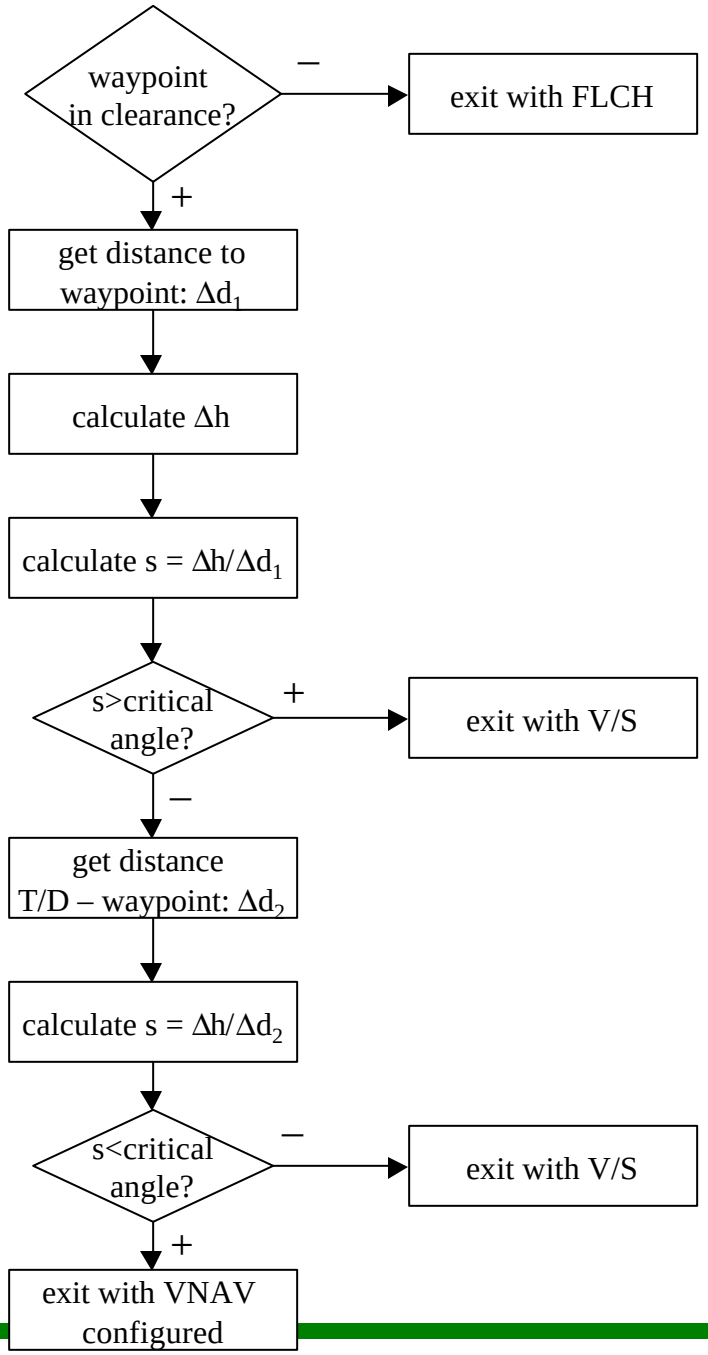


Normative Plan for Implementing Clearances



GMU

Elaborate Clearance / Select Method



GMU

Example NGOMSL Method

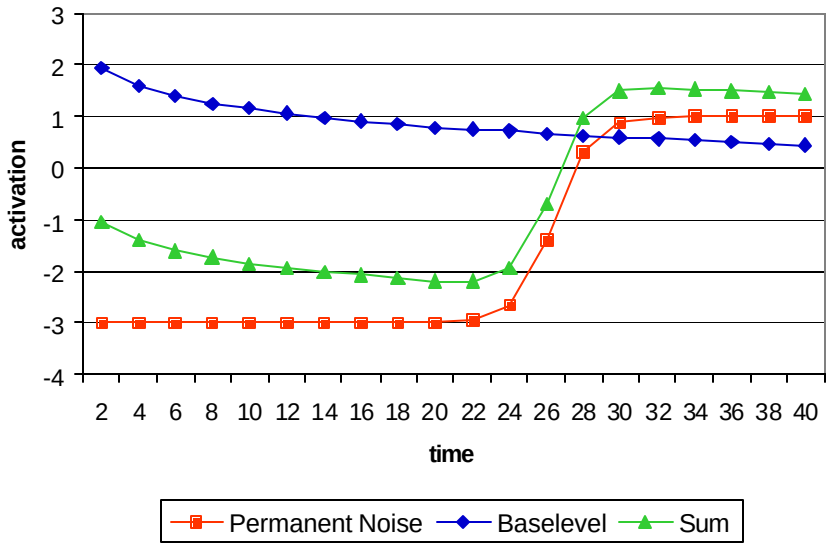
MFG: Change altitude using V/S wheel (Assume VNAV, ATT, AP engaged)

Step 1	Verify FMA shows ALT on pitch mode
Step 2	AG: Change MCP Alt
Step 3	Rotate V/S wheel up (for climb) or down (for descent) to set rate of descent
Step 4	Check V/S button is activated
Step 5	If not activated, push the V/S button
Step 6	Monitor for alt capture and subsequent hold
Step 7	Return with goal accomplished

Intentions

- Some methods contain special steps that form intentions (deferred actions)
- Intentions are represented by chunks of type main-goal
- Goals that represent intentions compete with all other goals for retrieval
- Preliminary solution for suppressing intentions for a while: adding permanent noise

```
(clrc-1 isa step type 'meth
 operator m-encode-clearance)
(clrc-2 isa step type 'sr)
(clrc-3 isa step type 'intend
 operator m-check-success)
(clrc-4 isa step type 'intend
 operator m-waypoint-close)
```

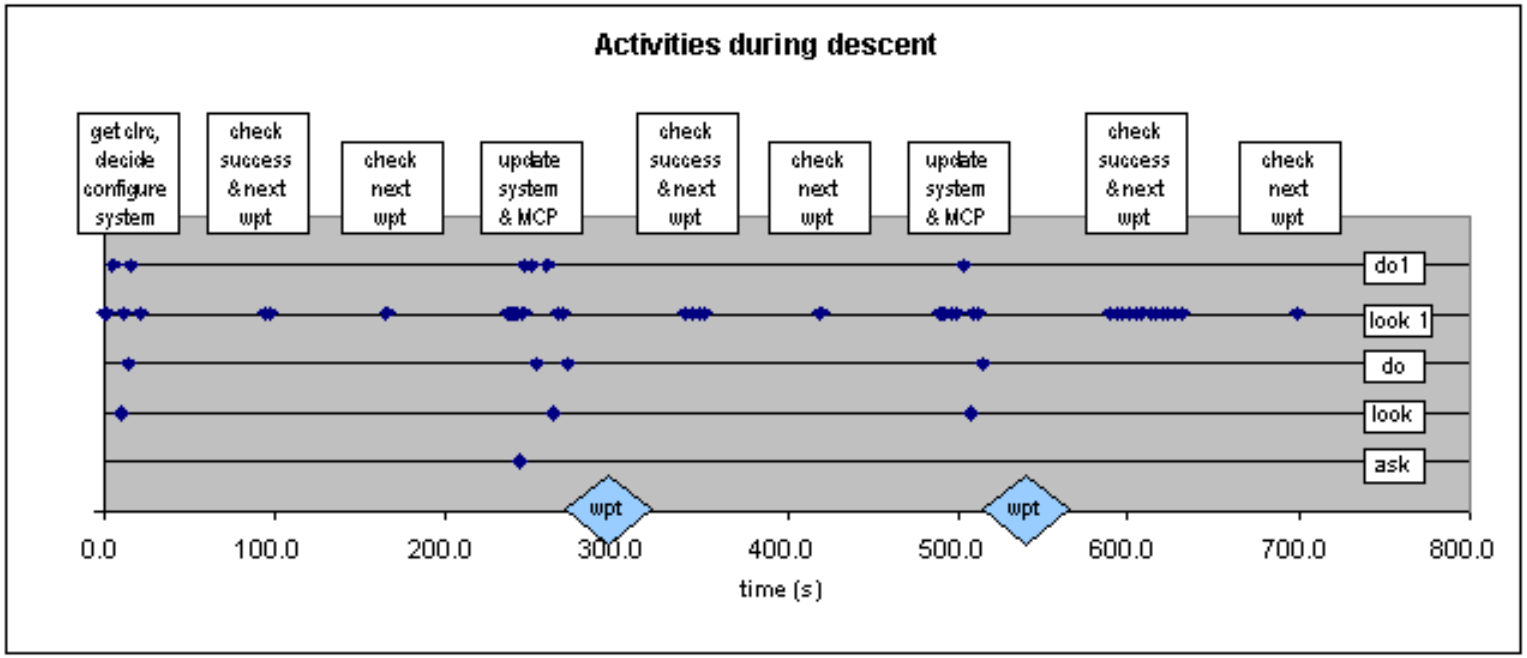


⇒ Important point here: the existence of the problem, not the solution

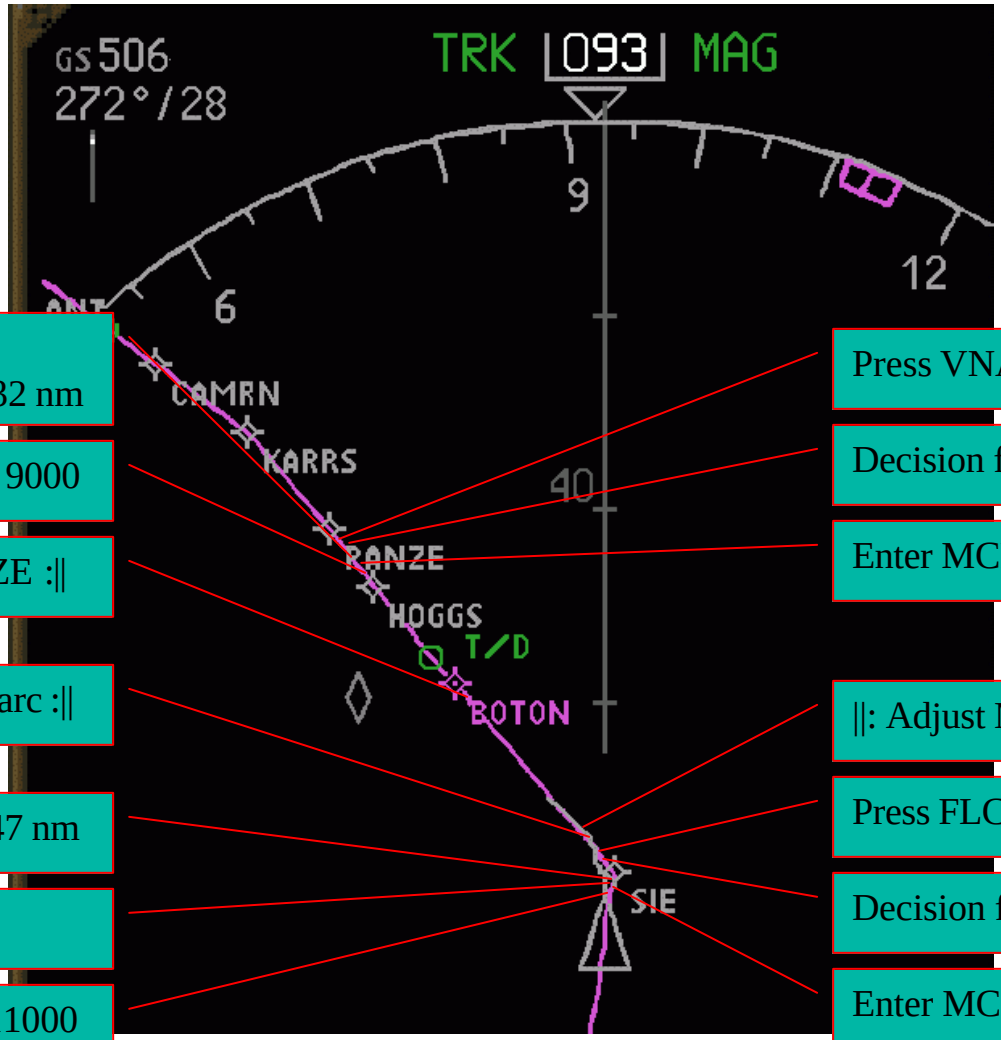
GMU

Model Behavior (I)

The scenario begins with a descent clearance; the model flies the simulated Boeing 747-400 from 23000 ft down to 10000 ft



Model Trace



Altitude: 12300
Distance to CAMRN: 32 nm

Next alt. Restriction: 9000

||: Check distance to PANZE :||

||: Check green arc :||

Distance to PANZE: 47 nm

Altitude: 23000

Clearance: PANZE 11000

Press VNAV

Decision for VNAV

Enter MCP-altitude 9000

||: Adjust MCP-speed :||

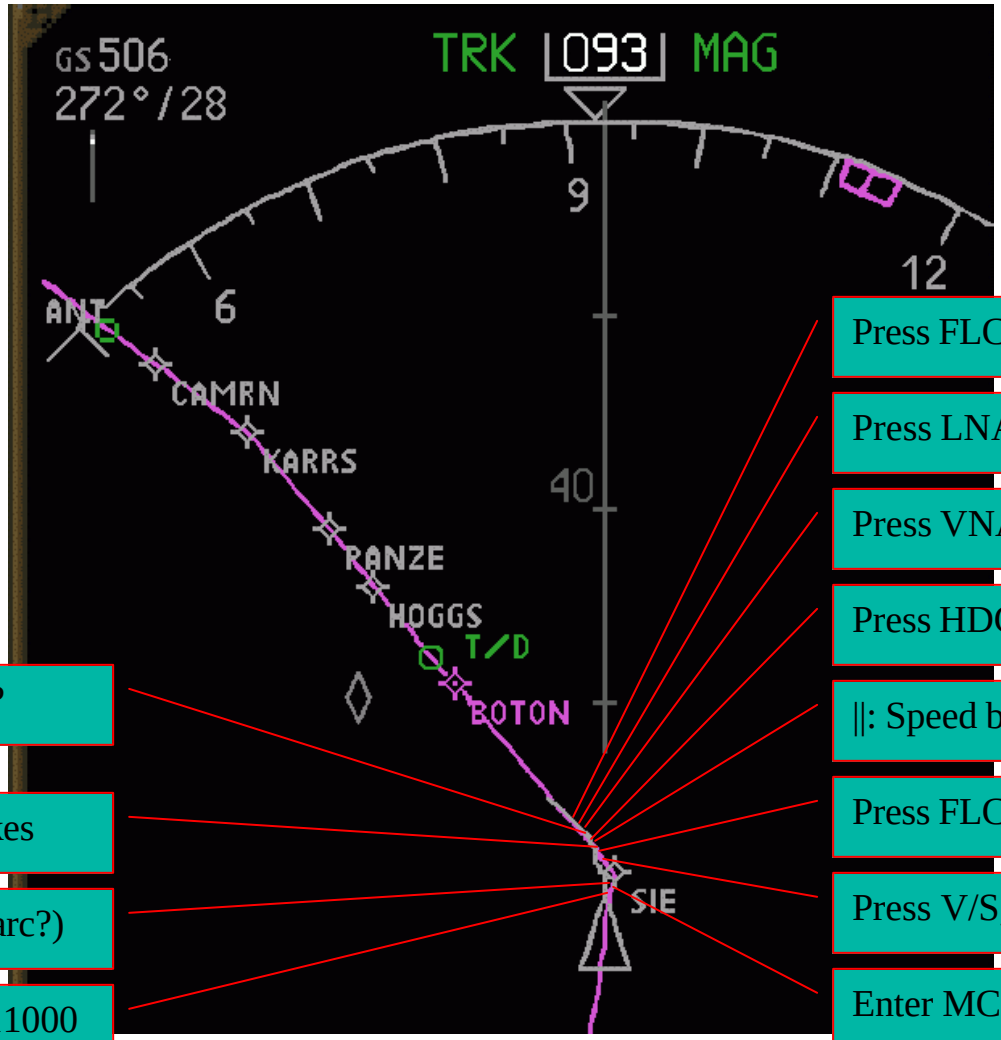
Press FLCH

Decision for FLCH

Enter MCP-altitude 11000

GMU

Subject Trace



?

PFD, Brakes

PFD, ND (altitude, green arc?)

Clearance: PANZE 11000

Press FLCH

Press LNAV

Press VNAV

Press HDG SEL

||: Speed brakes :||

Press FLCH

Press V/S, enter value

Enter MCP-altitude 11000

GMU

Single-Pilot Research

- Interaction with 747-400 FMS
- Study 1
  - Purpose was to inform single-pilot cognitive model
  - 5 UAL pilots flew a desk-top simulator for 2 legs
  - Eye-track, verbal protocol data collected
- Findings
  - Differences in scan strategies
  - During cued recall, pilots unable to recall FMAs

Single-Pilot Research

- Study 2 – Follow up
  - Goal is to further explore FMA confusion
    - Each pilot will act as Pilot Monitoring while watching videotape 747-400 simulator flying a descent scenario using FMS LNAV and VNAV
    - Descent scenario designed to emphasize uncommanded/ surprising mode changes
    - Videotape will be paused and knowledge measured at specific points in the scenario
  - Uses single-pilot model to explore possible interventions that will facilitate FMA understanding

Using the crew model

- Currently building the crew model
- Use model to identify factors affecting crew performance, e.g.:
  - task interruptions
  - high vs. low mental workload?
- Translate the effects shown by the crew model into guidance for assessment and training

Crew Processes

- Crew model should include some representation of crew-level processes relevant for automation
  - Information from interaction with airline Automation Philosophy and Training Group
    - Subject-matter experts on automation
  - Information from Jeff Beaubien's dissertation research
    - Normal sample of pilots from a major carrier during recurrent training

Crew Models

- Goals:
  - Model crew automation interaction
  - Improve assessment based on model
- Crew interaction model will focus on communication and actions
  - Two individual pilot models talking with each other
  - Leveraged from single-pilot automation model